Prime number generation using memetic programming

AUTHOR(S):

Mabrouk, Emad; Hernández-Castro, Julio César; Fukushima, Masao

Prime number generation using memetic programming

Emad Mabrouk[1], Julio César Hernández-Castro[2] and Masao Fukushima[3]

[1]*Dept. of Applied Mathematics and Physics, Graduate School of Informatics, Kyoto University,*

Kyoto 606-8501, JAPAN.

[1]*Dept. of Mathematics, Faculty of Science, Assiut University, Assiut, EGYPT.*

(hamdy@amp.i.kyoto-u.ac.jp)

[2]*School of Computing, University of Portsmouth, Buckingham Building, Lion Terrace,*

Portsmouth PO1 3HE, UK.

(Julio.Hernandez-Castro@port.ac.uk)

[3]*Dept. of Applied Mathematics and Physics, Graduate School of Informatics, Kyoto University,*

Kyoto 606-8501, JAPAN.

(Tel: +81-75-753-5519; Fax: +81-75-753-4756)

(fuku@i.kyoto-u.ac.jp)

Abstract: For centuries, the study of prime numbers has been regarded as a subject of pure mathematics in number theory. Recently, this vision has changed and the importance of prime numbers increased rapidly especially in information technology, e.g., public key cryptography algorithms, hash tables, and pseudorandom number generators. One of the most popular topics that attract attention is to find a formula that maps the set of natural numbers into the set of prime numbers. However, up to now there is no known formula that produces all primes. In this paper, we use a hybrid evolutionary algorithm, called the Memetic Programming (MP) algorithm, to generate mathematical formulas that produce distinct primes. Using the MP algorithm, we succeeded to discover an interesting set of formulas that produce sets of distinct primes.

Key Words: Hybrid Evolutionary Algorithm, Iterated Local Search, Memetic Programming, Prime Number

1. INTRODUCTION

A natural number grater than 1 is called a prime if it is only divisible by 1 and itself. The study of prime numbers and their properties have attracted mathematicians for several centuries. Questions related to prime numbers have puzzled mathematicians for many years, e.g., “is there a formula that maps the set of natural numbers into the set of primes?” Recently, several applications in the field of information technology have increased the importance of prime numbers, and changed the vision that classifies the study of primes as pure mathematics.

The Memetic Programming (MP) algorithm is a new evolutionary algorithm that hybridizes the well-known Genetic Programming (GP) algorithm, Koza [1], with some local search procedures over a tree space to intensify promising programs generated by the GP algorithm. The aim of this paper is to use the MP algorithm to generate some mathematical formulas which produce distinct primes for a set of consecutive integers.

The paper is organized as follows. In the next section, we introduce the MP algorithm briefly. In Section 3, we report the results of three experiments for generating formulas which produce distinct primes. Finally, conclusions make up Section 4.

2. MEMETIC PROGRAMMING

The MP algorithm is a hybrid evolutionary algorithm that searches for desirable computer programs as outputs. Computer programs treated in the MP algorithm are represented as trees in which leaf nodes are called terminals and internal nodes are called functions. Depending on the problem at hand, the user defines the domains of terminals and functions. In the coding process,

the tree structure of a solution should be transformed to an executable code. Fig. 1 shows two examples of MP programs represented as trees, along with their executable codes.

The main loop of the MP algorithm consists of two phases; diversification phase and intensification phase. In the diversification phase, the MP algorithm guarantees the diversity in the current population by using the GP strategy. Specifically, the MP algorithm selects some programs using a suitable selection strategy, and generates a new population from the current one by using crossover and mutation operators.

Fig. 2 illustrates an example of applying the crossover and mutation operators for some trees. In the intensification phase, the MP algorithm uses a set of local search procedures to intensify elite programs of the current population. These local search procedures will be described in the following subsection. Using these procedures, the Local Search Programming (LSP) algorithm will be introduced in Subsection 2.2, which is used in the intensification phase of the MP algorithm. Finally, the whole picture of the MP algorithm will be sketched in Subsection 2.3.

2.1 LOCAL SEARCHES OVER TREE SPACE

The local search procedures aim to generate new trees in a neighborhood of a given tree X. In this subsection, we discuss two types of local searches; static structure search and dynamic structure search, Mabrouk et al [2, 3]. Static structure search explores the neighborhood of a tree by altering its nodes without changing its structure. On the other hand, dynamic structure search changes the structure of a tree by expanding its terminal nodes or cutting its subtrees. Shaking search is used as a static structure search procedure, while grafting and pruning searches are

introduced as dynamic structure search procedures.

Shaking search generates a new tree $\tilde{X}$ from the current one X, by altering some nodes of X without changing its structure. The altered nodes are chosen randomly and replaced by alternative ones, i.e., a terminal node is replaced by a new terminal value and a function node is replaced by a new function of the same number of arguments as the original one. Grafting search generates a new tree $\tilde{X}$ from a tree X by replacing some of its terminals, chosen randomly, by branches of depth $\zeta \geq 1$, where these branches are generated randomly. In contrast to grafting search, pruning search generates an altered tree $\tilde{X}$ from a tree X by cutting some of its branches of depth $\zeta \geq 1$ and replacing them by new terminals, where these branches and terminals are chosen randomly. One may note that X and $\tilde{X}$ have different tree structures in case of applying the grafting or pruning procedures. Fig. 3 shows examples of generating a new tree $\tilde{X}$ from a tree X by applying shaking, grafting and pruning procedures. For more details see Mabrouk et al [2, 3].

2.2 LSP ALGORITHM

In this subsection, we introduce the LSP algorithm to discover the best program in the neighborhood of a program X. This algorithm uses the local search procedures described in the previous subsection to generate trial programs in the neighborhood of the program X. This process iterates until the termination condition is satisfied, and then the algorithm returns the best program found. Fig. 4 shows the flowchart of the LSP algorithm, where it returns the original program X in case of no improvement.

2.3 MP ALGORITHM

The main target of the MP algorithm is to improve the results of the GP algorithm by performing a local search for some promising programs. If the search process succeeds to reach the area near an optimal solution, then a simple local search algorithm can capture that optimal solution easily. Fig. 5 shows the flowchart of the MP algorithm that behaves like the GP algorithm if the LSP algorithm fails to improve the selected programs. However, in this case the MP algorithm will be more costly than the GP algorithm, because of the computational effort spent through the intensification phase.

Each program treated in the MP algorithm consists of one or more gene(s), where a gene represents a sub-tree consisting of terminal and function nodes. Genes in a program are linked together by using a suitable linking function. The addition function "+" is used as the linking function in this paper. To generate a gene in the initial population, we generate a temporary random gene consisting of two parts, head (functions and terminals) and tail (terminals only). Then, we adjust the final form of the gene by deleting unnecessary elements, based on the functions and terminals that are generated randomly within the gene. For more details see Mabrouk et al [2, 3].

Once the initial population is generated, it will be evolved and improved using the MP operations; i.e., crossover, mutation, shaking, grafting and pruning. For each problem to solve, the function set, the terminal set, the set of representation parameters, the set of search parameters, and the fitness function must be determined before calling the algorithm. The set of representation parameters contains the head length *hLen* of an initial gene, the maximum length *MaxLen* of a

gene, the number of genes *nGenes* of each program. On the other hand, the set of search parameters, which guide the algorithm during the search process, consists of the population size *nPop*, the number of generations *nGnrs*, the number of programs *nLs* that are selected to apply local search procedures, the number of trial programs *nTrs* that are generated in the neighborhood of a program using a local search procedure, and the maximum number of non-improvements *nFails*. In particular, *nFails* is used to terminate the LSP algorithm. For more details about these parameters see Mabrouk et al [2, 3].

3. NUMERICAL EXPERIMENTS

In this section we report the results of three different experiments for the MP algorithm to generate formulas that produce primes. The parameter values for the MP algorithm during all experiments are *hLen* = 3, *MaxLen* = 40, *nGenes* = 3, *nLs* = 4, *nTrs* = 4, *nFails* = 1, *nPop* = 100 and *nGnrs* = 100. In addition, the selection strategy for the diversification phase is the tournament selection of size 4. The fitness value for each program is computed as the maximum number of consecutive integers in the interval [–100, 100] for which the program produced distinct primes.

3.1 POLYNOMIALS

In this experiment, we used the set of binary functions {+, –, *} as the function set, i.e., each program generated by the MP algorithm represents a polynomial. In addition, we used {*x*, 2, 3, 5, 7, 9} as the terminal set, where *x* is an integer. We performed 1000 independent runs for the MP algorithm, and we got a number of polynomials with the fitness values up to 40.

Table 1 shows some of polynomials which generated by the MP algorithm. The first three polynomials in Table 1 have already been found in the literature. Specifically, the first two polynomials are the Euler and Legendre polynomials, and the third one is the polynomial generated by the Cartesian GP (CGP) algorithm, Walker [4]. During our experiments, these three polynomials were found frequently. To the best of the authors' knowledge, the other polynomials seem to be new polynomials.

In the literature, researchers consider the first three polynomials in Table 1 to be different polynomials. However, all of these polynomials produce the same set of primes for different values of the independent variable x. Specifically, one can generate those entire polynomials one after another by using $x := x - \lambda$ for some integers λ. On the other hand, the last three polynomials in Table 1 produce different sets of distinct primes for different sets of consecutive integers. Therefore, we consider these three polynomials to be the best results for the current experiment since all of them are different and independent.

3.2 RATIONAL FUNCTIONS

We performed another experiment to find formulas that produce primes with fitness values greater than 40. In this experiment, we modified the function set in the previous experiment to include the protected division operator %, where $x \,\%\, y = 1$ if $y = 0$, and $x \,\%\, y = x / y$ otherwise. In this case, programs of the MP algorithm will produce real values. Therefore, we let the nearest integer less than or equal to the produced real value be the output of the program. Using the new function set, we got several new formulas that produce up to 42 distinct primes for a set of

consecutive integers, for example, $|\lfloor(-8x^3 + 69x^2 - 461x - 176) / (8x + 3)\rfloor|$, with the fitness value 42.

3.3 COMPOSITION FUNCTIONS

Since we have already got new independent polynomials that can generate different sets of distinct primes, we can use these polynomials to composite new formulas. In this experiment, the output of a program evolved by the MP algorithm is expressed as a linear composition of its genes with some independent polynomials that produce distinct primes. Suppose that G_1, G_2 and G_3 are the genes of a program evolved by the MP algorithm. Then, the output formula of this program is composed as $f(x) = \lfloor G_1 \rfloor * P_1 + \lfloor G_2 \rfloor * P_2 + \lfloor G_3 \rfloor * P_3$, where P_1, P_2 and P_3 are independent polynomials. Using this strategy, we got new formulas that produce distinct primes up to 59, for example, $|\lfloor 7 / (81x + 27) \rfloor (x^2 + x + 41) + \lfloor 9 / (5 - 45x) \rfloor (8x^2 - 22x - 647)|$, with the fitness value 59.

4. CONCLUSIONS

The MP algorithm has succeeded to generate several new formulas that produce sets of distinct primes. Some of the new formulas are polynomials that are able to produce up to 40 distinct primes for a set of consecutive integers. Other rational functions are also generated and they are able to produce up to 59 distinct primes for a set of consecutive integers.

REFERENCES

[1] Koza JR (1992) Genetic programming: On the programming of computers by means of natural selection. MIT Press, Cambridge

[2] Mabrouk E, Hedar A, Fukushima M (2008) Memetic programming with adaptive local search using tree data structures. In: Chbeir R et al (eds.) Proceedings of the 5th International Conference on Soft Computing as Transdisciplinary Science and Technology (CSTST08), Cergy-Pontoise, Paris, France, October 27–31, 2008, pp. 258–264

[3] Mabrouk E, Hedar A, Fukushima M (2010) Memetic programming algorithm with automatically defined functions. Technical Report 2010-015, Department of Applied Mathematics and Physics, Kyoto University, Japan

[4] Walker JA, Miller JF (2007) Predicting prime numbers using Cartesian genetic programming. In: Ebner M et al. (eds.), Proceedings of the 10th European Conference on Genetic Programming (EuroGP), Valencia, Spain, April 11–13, 2007, pp. 205–216

Table1 Polynomials generated by the MP algorithm to produce distinct primes

Polynomial	Fitness	x
$x^2 - x + 41$	40	{1,…,40}
$x^2 + x + 41$	40	{0,…,39}
$x^2 - 3x + 43$	40	{2,…,41}
$9x^2 + 33x + 71$	40	{-28,…,11}
$4x^2 - 50x + 197$	40	{-13,…,26}
$8x^2 - 22x - 647$	40	{-19,…,20}

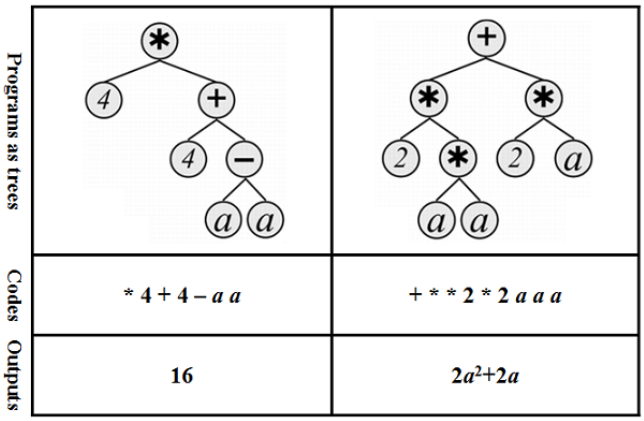

Fig.1 Examples of MP representation

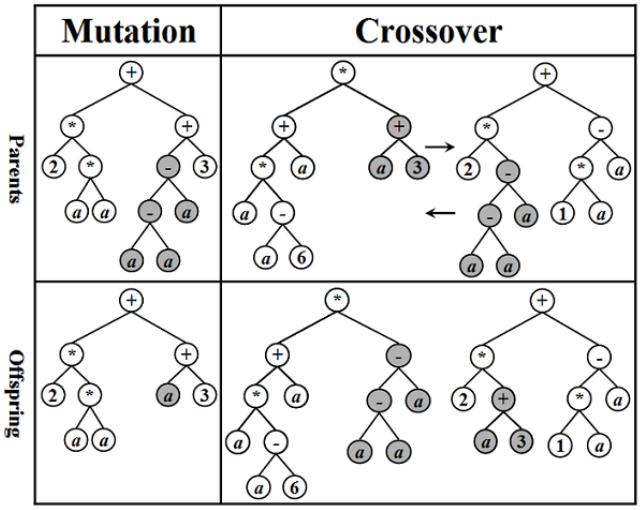

Fig.2 Generating a new offspring using mutation and crossover operators

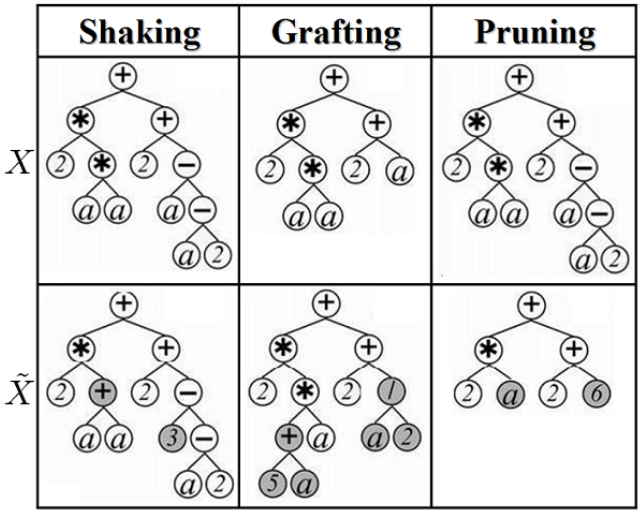

Fig.3 Generating new trees using shaking, grafting and pruning procedures

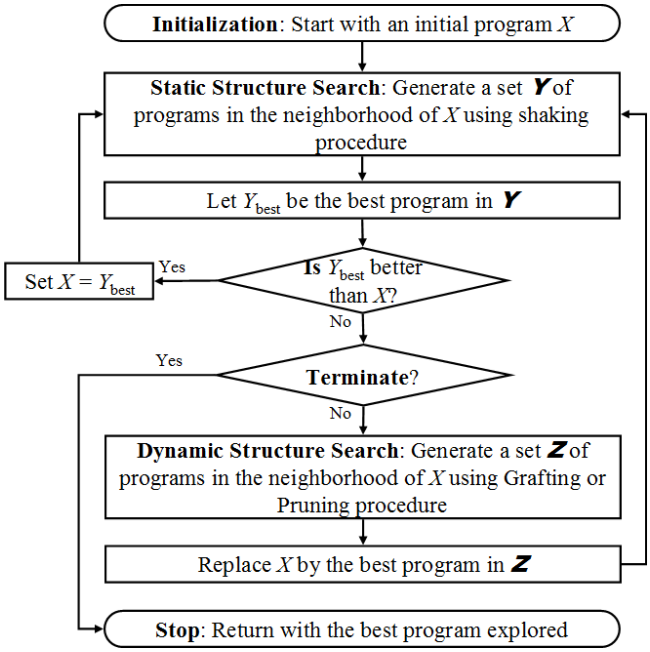

Fig.4 The flowchart of the LSP algorithm

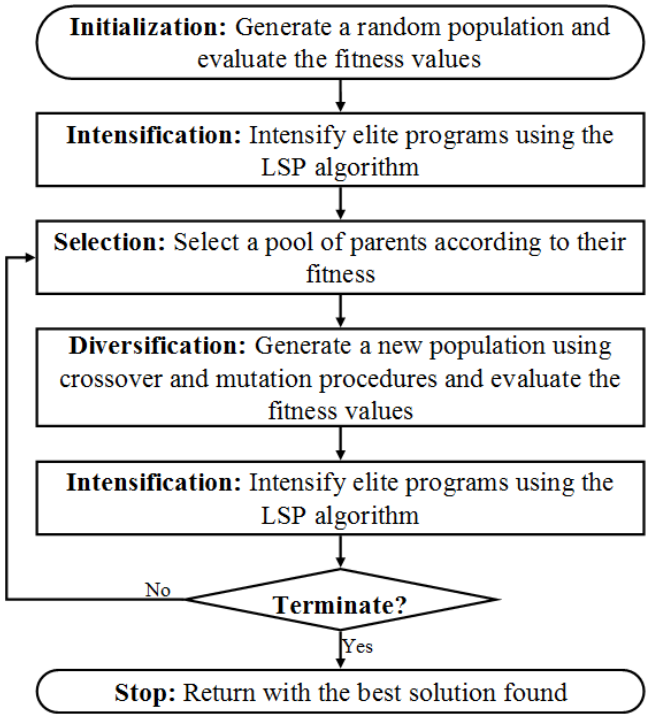

Fig.5 The flowchart of the MP algorithm